Adaptive Deep Q-Networks for Decision Making in Non-Stationary Environments: A Case Study with the Wisconsin Card Sorting Test

Dieu-Donné Fangnon [1,2] **Eduardo H. Ramirez-Rangel** [3]

1 African Masters of Machine Intelligence (AMMI)

dfangnon@aimsammi.org

2 African Institute for Mathematical Sciences (AIMS)

3 Ensitech USA LLC / Tecnologico de Monterrey

eduardo.h.ramirez@tec.mx

Abstract

Adaptive learning in dynamic environments presents significant challenges in reinforcement learning, especially when the environment's rules change unpredictably. This study introduces an enhanced Deep Q-Network (DQN) architecture designed to effectively handle non-stationary environments, with a focus on the Wisconsin Card Sorting Test (WCST) as a representative problem. Unlike conventional DQN approaches that struggle with rule changes, our method integrates rule context directly into the state space, allowing the agent to adapt its strategy dynamically. We detail the modifications to the traditional DQN architecture, which include extending the input state with rule-specific information and adapting the network to process these extended states efficiently. Experimental results demonstrate that our adaptive DQN significantly outperforms traditional DQN models in terms of flexibility and accuracy in the WCST, showcasing its potential to generalize across different types of non-stationary environments. This approach not only enhances the agent's performance but also contributes to the broader application of deep reinforcement learning in complex, changing systems.

Keywords: Wisconsin Card Sorting Test (WCST), Non-Stationary Environment, Deep Reinforcement Learning

1 Introduction

Adapting to dynamic environments is a cornerstone challenge in the field of reinforcement learning (RL). Traditional RL methods often assume stationary environments, where the underlying dynamics and reward structures do not change over time. However, many real-world applications, from financial markets to adaptive control systems, are inherently non-stationary, where rules and conditions can shift unpredictably. This non-stationarity can severely hinder the performance of conventional RL agents due to their inability to adapt to new environmental conditions without retraining from scratch.

The Wisconsin Card Sorting Test (WCST) is a classical psychological test that exemplifies a non-stationary environment. In the WCST, subjects must discover the sorting rule (e.g., color, shape, or number) based on feedback and adapt their strategy when the rule changes without warning. This test has been a benchmark in cognitive psychology to assess executive function and flexibility in

task switching, and it poses unique challenges for RL algorithms due to its rule-changing dynamics. The Wisconsin Card Sorting Test serves as a psychological assessment tool aimed at gauging cognitive flexibility in individuals exhibiting decision-making deficits, often indicative of frontal lobe dysfunction.It has long stood as a benchmark in cognitive psychology, challenging individuals' cognitive flexibility, working memory, and problem-solving abilities [1]. Introduced by Heaton in 1981 [2], the WCST requires the participant to sort a set of cards according to implicit rules and based on the limited corrective feedback provided by the examiner. The participant's responses can be analyzed to produce separate indices of sources of difficulty on the test. The value of the WCST, therefore, lies in its sensitivity for detecting and characterizing aspects of executive dysfunction. The WCST consists of two identical sets of 64 response cards that differ on three possible perceptual sorting dimensions: color (red, blue, yellow, or green), form (crosses, circles, triangles, or stars), and number of figures (one, two, three, or four). Four target cards that represent the range of dimensions (one red triangle, two green stars, three yellow crosses, and four blue circles) are placed in a prescribed order in front of the participant. On each trial, a response is requested that assigns the current target card to one of four simultaneously presented key cards. The task is selecting on each trial the key card that shares the feature with to be prioritized rule contingent target feature [3]. This matching is done in reference to four predefined cards. Crucially, the rule governing the matching process must be deduced through trial and error, and it may change unexpectedly (see fig. 1). The test evaluates the subject's capacity to adapt to such rule changes. Traditionally, performance on the WCST has been analyzed through manual scoring methods or basic reinforcement learning (RL) algorithms. However, these approaches often struggle to capture the dynamic and non-stationary nature of the WCST, which involves rule changes that challenge the adaptability of the participant. Over the years, researchers have employed various approaches to tackle this intricate problem, ranging from heuristic-based strategies to rule-based models. However, despite the diverse methodologies employed, a notable gap exists in the exploration of Deep Reinforcement Learning (Deep RL) techniques to enhance WCST performance.

Traditional Deep Q-Networks (DQN), introduced by Mnih et al. [4], revolutionized the field of RL through their ability to combine deep neural networks with Q-learning, achieving superhuman performance in various Atari 2600 games. However, the standard DQN architecture also struggles with tasks like the WCST where the environment's rules can change within a single episode. These challenges are primarily due to the static nature of the learned policies in standard DQN frameworks, which do not account for the environment's dynamic aspects. Recent advances have sought to extend DQN architectures to better handle such environments. Techniques such as meta-learning, which aims to train models on a variety of tasks that enable fast adaptation to new tasks (Finn et al., 2017 [5]), and modular networks that dynamically reconfigure their structure in response to environmental changes, offer promising directions. However, these methods can be complex and computationally intensive.

In this paper, we propose a novel approach to enhancing DQN for dynamic environments by integrating the rule context directly into the state representation. This method allows the DQN agent to maintain awareness of the current rule as part of its decision-making process, thereby adapting its policy in response to changes without the need for retraining or human intervention. We demonstrate the effectiveness of our approach through extensive experiments on a simulated WCST environment, comparing our results against traditional DQN models to highlight the benefits of our method.

The paper is structured as follows: Section 2 reviews related work in the domain of adaptive reinforcement learning and its application to non-stationary environments. Section 3 and 4 describes our enhanced DQN architecture and the modifications made to adapt it to the WCST. Section 5 presents our experimental setup, results, and a discussion of our findings. Finally, Section 6 concludes the paper with a summary of our contributions and potential avenues for future research.

2 Related work

The Wisconsin Card Sorting Test (WCST) has been a subject of extensive study in both neuroscience and artificial intelligence, serving as a benchmark to assess the ability of systems to learn and adapt to changing rules and conditions. Several approaches have been undertaken to tackle the WCST, each offering insights into handling non-stationary problems with varying degrees of success.

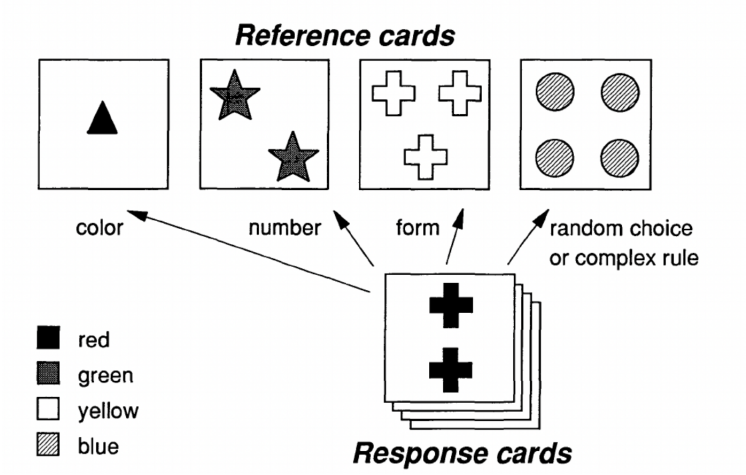

Figure 1: Material used in the Wisconsin Card Sorting Test (adapted from Milner. 1963). The patient must place each response card under 1 of the 4 reference cards, and is then told by the experimenter whether the choice was right or wrong. On the basis of this feedback, the patient must discover the correct sorting rule: color, number, or form [6]

Early work in adapting RL to non-stationary environments often involved modifying traditional algorithms to better handle changing conditions. Methods such as learning rate adjustments (Sutton, 20216 [7]) and sliding window techniques (Kaelbling, 1993 [8]) were proposed to update the agent's knowledge base continuously without discarding the relevance of past learning. More sophisticated approaches, like the use of ensemble techniques (Dietterich, 2002 [9]), have been utilized to maintain multiple hypotheses of potential environment dynamics simultaneously.

With the advent of Deep Q-Networks (DQN) by Mnih et al. [4], there was a shift towards integrating deep learning with RL, leading to significant performance gains in several benchmark tasks. However, the application of DQN to non-stationary environments exposed its limitations due to the static nature of its learning architecture. Researchers have proposed several enhancements to DQN for dynamic settings, such as the use of recurrent neural networks (RNNs) to capture temporal dependencies within the environment (Hausknecht and Stone, [10]) and the incorporation of external memory units to store past experiences selectively.

Steinke et al. [11] proposed a hybrid method combining machine learning algorithms and cognitive theories to address the challenges posed by the WCST. Their approach, while innovative, relied on predefined rule sets and lacked the continuous adaptability required for non-stationary environments. Our method improves upon this by using a DQN that adjusts its strategy in response to real-time changes, reflecting more closely the demands of continuously evolving tasks.

Qin et al. [12] introduced a technique for non-stationary representation learning in sequential linear bandits, which could be beneficial for tasks like WCST that require adaptation to evolving rules. Their algorithm efficiently transfers learned representations across changing tasks, which is theoretically appealing. However, unlike our approach, it does not integrate these capabilities within a neural framework, potentially limiting its applicability in deep learning contexts where direct integration with neural networks can enhance performance and adaptability.

Our method synthesizes insights from these diverse approaches, embedding rule adaptability within the neural architecture itself. This not only simplifies the model's structure by eliminating the need for external rule management mechanisms but also enhances its responsiveness to changing environmental conditions. By integrating the rule context directly into the state inputs of the DQN, our approach offers a robust solution to the challenges of non-stationary environments, demonstrating superior performance and flexibility compared to traditional models.

3 RL model for non-stationary environments

3.1 Reinforcement learning

Reinforcement learning is one of the machine learning methods, which learns to perform serial actions according to situations in order to maximize the reward signal by trial-and-error search [13]. Unlike supervised learning which learns from a training set of labeled examples, reinforcement learning learns from its own experience of interaction with all kinds of situations; in addition, reinforcement learning differs from unsupervised learning as it aims to maximize the reward signal rather than find the hidden structure of the data [33]. In recent decades, reinforcement learning has been proved effective in solving decision-making problems.

Most of the reinforcement learning problems can be modeled in a mathematical form of the Markov decision process. In MDP, an agent learns and makes decisions; other things the agent interacts with are called environment [13]. At decision epoch t, the agent chooses an action a_t according to environment state s_t. At the next epoch, the agent receives a reward r_t and a new state s_{t+1} (see the stationary case in Fig. 1). Thus, MDP can be represented by a tuple of $\langle S, A, P, R\rangle$, where S and A are the state and action spaces. $P : S \times A \times S \to [0, 1]$ is the transition probability function. It defines the probability distribution of next state s' at $t + 1$ the system evolves into, given the current state s and action a (see Eq. 1). $R : S \times A \to \mathbb{R}$ is the reward function. It models the numerical reward yielded by applying action a to state s, i.e., $R \sim r(s, a)$.

$$p\left(s' \mid s, a\right) = \Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\} \tag{1}$$

Reinforcement learning algorithms can also be categorized into value-based methods (e.g., Q-learning) and policy-based methods (e.g. policy gradient). These algorithms have been combined with deep neural networks for more complicated control problems.

3.2 Markov Decision Process and RL algorithms in non-stationary environments

Technically speaking, the non-stationarity of the building environment mainly refers to the fact that the state transition probability P and reward function R are non-stationary. They are changing along with some internal or external factors. To be more concrete, even though the instant states (e.g., indoor air temperature) are similar in different periods, the building state evolutions can be different. It is insufficient to describe such a control problem as a stable MDP. Thus, [14] defined MDP in non-stationary environments as below:

Given a family of MDPs $\{M_\theta\}_{\theta \in \mathbb{N}_+}$, where θ takes values from a finite index set $\Theta, M_\theta = \langle S, A, P_\theta, R_\theta\rangle$ with the same state and action spaces and different transition probability distributions and reward functions. Each MDP in the family is called a context.

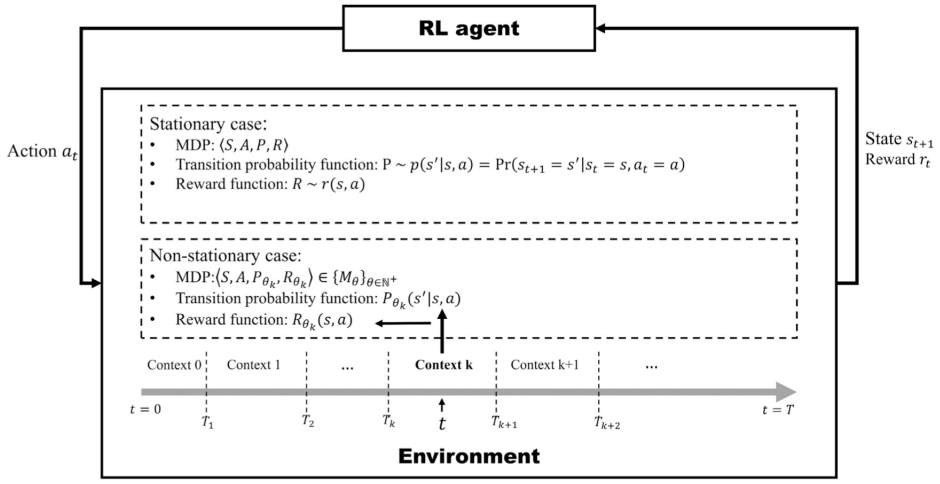

Figure 2: Reinforcement learning in stationary and non-stationary environments. [15]

Fig. 2 demonstrates the non-stationary case of environments in reinforcement learning.

In the literature, there have been some pioneer works addressing RL in non-stationary environments. One approach aims to minimize the performance criterion called regret (see Eq. 2), which refers to the difference between the expected optimal rewards collected over a finite horizon T from a start state s_0, and those produced by the current policy. [16] proposed UCRL2 algorithm, a variant of Upper Confidence Reinforcement Learning (UCRL) proposed by [17]. UCRL2 adopts this approach and gives a sublinear regret upper bound by finding the optimal MDP from a set of plausible MDPs for the current environment and its corresponding optimal policy. A set of plausible MDPs is built by estimating the transition probability function and the reward function. UCRL2 restarts learning when it exceeds the diameter of the current MDP (see Eq. 3), where M is the environment context and T is the timestep of the first arrival of s' from s), which however will discard all the former estimations.

$$Regret = V_T^* (s_0) - \sum_{t=0}^{T-1} R (s_t, a_t) \tag{2}$$

$$D_M = \max_{s \neq s'} \min_{\pi : S \to A} \mathbb{E} [T (s' \mid s, \pi)] \tag{3}$$

3.3 Non-Stationary MDP Formulation for WCST

Given a set of possible rules $\Theta = \theta_1, \theta_2, \theta_3$ that dictate the matching criterion (e.g., color, shape, number), the environment can be described by a family of MDPs $M_\theta, \theta \in \Theta$. Each MDP, $M_\theta = \langle S, A, P_\theta, R_\theta \rangle$, shares the same state and action spaces but has different transition probabilities and reward functions depending on the rule θ in effect.

- **Transition Probabilities and Rule Dynamics:** The state transition probabilities are influenced by the active rule, reflecting how the rule change affects the environment's dynamics:

$$P (s_{t+1} = s' \mid s_t = s, a_t = a, \theta_t = \theta) = \begin{cases} P_{\theta_0} (s' \mid s, a) & \text{if } t < T_1 \\ P_{\theta_1} (s' \mid s, a) & \text{if } T_1 \leq t < T_2 \\ \vdots \end{cases}$$

  Where θ_t represents the rule active at time t, and T_i denotes the timestep at which a rule change occurs, transitioning from one context (rule) to another.

- **Reward Function:** Similarly, the reward function is contingent upon the current rule, underscoring how rewards are aligned with the rule-based objectives:

$$R(s, a, \theta) = \begin{cases} R_{\theta_0}(s, a) & \text{if } t < T_1 \\ R_{\theta_1}(s, a) & \text{if } T_1 \leq t < T_2 \\ \vdots \end{cases}$$

- **Rule Change Mechanism:** The rule change is modeled as a stochastic process governed by a probability $\rho(\theta' \mid \theta)$, determining the likelihood of transitioning from rule θ to θ':

$$\theta_{t+1} \sim \rho (\theta' \mid \theta_t)$$

This mathematical framework allows us to explicitly incorporate the rule dynamics into the learning process of the DQN agent, facilitating the development of strategies that are robust to the changing conditions of the task.

4 Methodology

4.1 Deep Q Network

Deep Q Network (DQN) combines Q-learning and deep neural network. As mentioned before, Q-learning is a value-based method. It learns a lookup table called a Q-table to store Q-values for

specific state-action (s, a) pairs. Thus, Q-learning is also called tabular Q-learning. The Q-values are updated by the experience tuple (s_t, a_t, r_t, s_{t+1}) every iteration the agent interacts with the environment. The update rule is based on the Bellman equation:

$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right)$

where α is the learning rate and γ is the discount factor. Tabular Q-learning methods suffer from explosive memory for tracking tremendous state-action pairs when faced with MDP with large size.

- **Online DQN from ACME**
  Deep Q-Networks (DQN) ([4, 18]Mnih et al, 2013, 2015) is a modern variation on the classical Q-Learning algorithm ([19] Watkins and Dayan, 1992), which uses a deep neural network to estimate the Q-function in discrete action spaces. DQN learns Q_ϕ by iteratively applying the recursive update described in the previous section and calculated by minimizing the loss in Equation 4 (with actions selected by the greedy policy). One optimization implemented by this algorithm relies on the fact that that it is specifically targeted at discrete action spaces. Rather than implement Q_ϕ as a function of both observations and actions, DQN actually learns a vector-valued function of the observations where the a-th output is the value of the associated action, i.e. $Q_\phi(s,a) = \left[Q_\phi(s)\right]_a$. This entire vector can then be calculated using a single pass through the network. In terms of data-generation DQN relies on an ϵ-greedy behavior policy for exploration, i.e. one in which with probability $\epsilon < 1$ a uniformly random action is selected and otherwise the greedy, maximum valued action is taken. As done in many other algorithms the value of ϵ is frequently annealed towards zero as learning proceeds. Data generated by this process is added to replay as n-step transitions which are used to form n-step returns during learning. In their implementation of DQN, and following in the spirit of Rainbow DQN ([20] Hessel et al., 2018), their also include a number of recent enhancements including Double Q-Learning, dueling networks, and prioritized experience replay using the TD error as the priority.

$$L(\phi) = \frac{1}{2}\mathbb{E}_{\rho_\pi}\left[\left(Q_\phi\left(S_t, A_t\right) - Y_t\right)^2\right] where Y_t = R_t + \gamma \mathbb{E}_{A\sim\pi(\cdot|S_{t+1})}\left[Q_{\phi'}\left(S_{t+1}, A\right)\right] \quad (4)$$

4.2 Enhancements to the DQN Architecture for WCST (Our WCST NS-DQN)

- **Integration of Rule Context into the State Space:** The state vector in our DQN model is augmented by appending the current rule as an additional feature. This inclusion transforms the state space S into an extended space S', where each state $s' \in S'$ is represented as:

  $$s' = (s, r)$$

  Here, s denotes the original state from the environment, and r is an integer representing the current rule. This augmentation allows the network to be sensitive to the context set by the current rule, enabling it to adapt its policy based on the rule's constraints and objectives.

- **Architectural Changes:** To process the augmented state space effectively, we have made modifications to the architecture of the neural network used in the DQN. The input layer is resized to accept the dimensionality of S', and the architecture is adapted as follows: - The input layer now has $n + 1$ units, where n is the number of features in the original state space, and the additional unit is for the rule feature. - Additional neurons in subsequent layers to manage the increased complexity and maintain the network's capacity for learning detailed patterns.

The core of the DQN learning process remains the optimization of the Q-value function, which estimates the value of taking an action a in a given state s under policy π. The Q-values are updated using the Bellman equation as follows:

$$Q_{\text{new}}\left(s,a\right) = Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right]$$

Where: α is the learning rate, r is the reward received after taking action a in state s, γ is the discount factor, s' is the new state after action a is taken, $\max_{a'} Q\left(s', a'\right)$ represents the highest Q-value achievable from the new state s'.

By integrating the rule context into the state space and adjusting the DQN architecture, our enhanced model not only responds adaptively to rule changes within the WCST but also improves decision-making accuracy in dynamic, non-stationary environments. These capabilities are essential for tasks where rules and objectives evolve over time, necessitating a flexible and context-aware learning approach.

5 Experiments

5.1 WCST Environment Description

The Wisconsin Card Sorting Test (WCST) presents a dynamic environment mimicking a psychological test used to evaluate executive function and cognitive flexibility. Participants must sort cards by different attributes color, shape, or number without being explicitly informed of the current sorting rule. The environment simulates this by requiring agents to determine the sorting rule from environmental cues and rewards.

The environment is defined by the following components:

- **Action Space:** The agent can take one of four discrete actions at each step, each corresponding to selecting one of four possible card categories for sorting.
- **State Space:** A multi-dimensional vector represents the observable attributes of the current card, including color, shape, and number. These features are numerically encoded and form the input state vector for the agent's decision-making process.
- **Rule Dynamics:** The underlying rule that dictates the correct card category changes unpredictably after a series of successful sorts. This rule change is not communicated directly to the agent; instead, the agent must deduce the change from the rewards and penalties received from its actions.

Observation and Reward Structure: At each step, the agent receives an observation that includes the features of the presented card and a reward signal. The reward signal is binary, with a positive reward for a correct match according to the current, but unknown, rule, and a negative reward for an incorrect match. The immediate change in reward following a sequence of correct actions signals a rule change, prompting the agent to adapt its strategy.

This WCST environment is an abstract simulation designed to challenge and train cognitive adaptability. Success in this environment relies on the agent's ability to quickly adapt to changing rules—a capacity that stands at the core of flexible, intelligent behavior and is crucial for navigating non-stationary, real-world scenarios.

5.2 Agent Evaluation and Comparative Analysis

To validate the performance of our Non-Stationary Deep Q-Network (NS-DQN) agent, we benchmarked its performance against a baseline DQN agent and a random agent. The key metrics for evaluation were cumulative reward, accuracy, and error rate across multiple episodes.

5.2.1 Evaluation Metrics:

- **Cumulative Reward:** The sum of rewards obtained by the agent throughout an episode, serving as a measure of overall effectiveness in rule identification and adaptation.
- **Accuracy:** The proportion of the agent's actions that resulted in positive feedback, reflecting the ability to correctly identify and sort cards based on the current rule.
- **Error Rate:** The frequency of incorrect actions taken by the agent, indicating the challenges faced in rule discernment and execution of the task.

5.2.2 Agent Descriptions:

- **Random Agent:** Acts without any learning or strategy, choosing actions uniformly at random, providing a baseline for the minimum expected performance.

- **Baseline DQN Agent:** Employs a standard DQN architecture without enhancements for non-stationary environments, offering a comparison against a well-established benchmark.
- **NS-DQN Agent:** Our proposed agent that integrates rule context into the state representation and adapts its neural network architecture to accommodate the dynamic nature of the WCST environment.

5.3 Results and Discussion

In this section, we evaluate the performance of each agent within the non-stationary environment, training them across three distinct episode counts until convergence is observed: 100, 500, and 1000 episodes.

The results depicted in the figures below (3, 4 & 5) highlight the individual performance of each agent, assessed by the cumulative rewards earned over the respective number of episodes. It is evident from the data that the baseline DQN agent significantly outperforms the random agent. However, our novel agent, designated as WCST NS-DQN, demonstrates superior performance over the baseline DQN in all episode intervals. These findings are corroborated by the comparative analysis presented in figure 6.

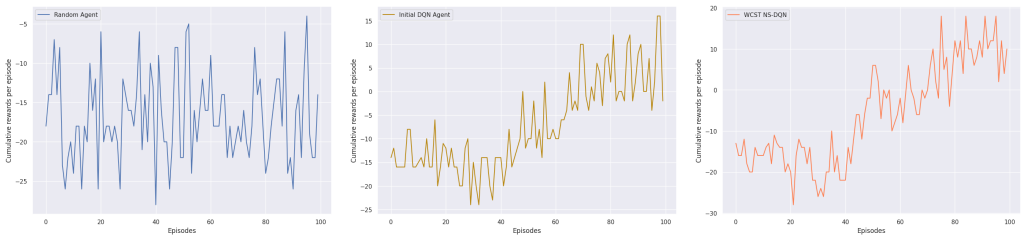

Figure 3: The performance of the three agents over 100 episodes in the non-stationary environment

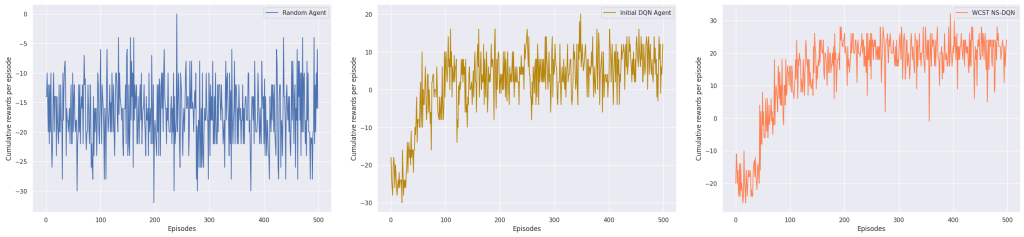

Figure 4: The performance of the three agents over 500 episodes in the non-stationary environment

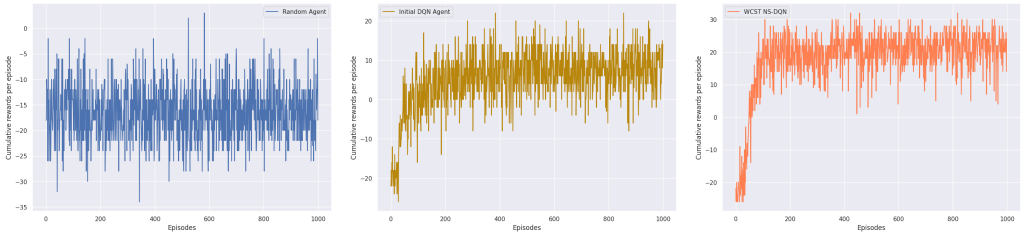

Figure 5: The performance of the three agents over 1000 episodes in the non-stationary environment

From the figure 6 below, the cumulative reward achieved by the WCST NS-DQN agent consistently surpassed that of the baseline DQN, as well as the random agent, which performed as expected with the lowest cumulative reward. Notably, the NS-DQN agent demonstrated a remarkable increase in accuracy and a significant reduction in the error rate when compared to its counterparts.

The baseline DQN agent showed limitations in adapting to the changing rules, as indicated by its moderate accuracy and higher error rate compared to the NS-DQN agent.

Table 1: Performance comparison of agents on WCST Over varying episodes.

		Random Agent	Baseline DQN Agent	WCST NS - Agent
n_episodes = 100	Accuracy	24.35	38.93	**41.14**
	Error Rate	75.65	61.07	**58.77**
	Number of wins	800	1289	**1375**
n_episodes = 500	Accuracy	24.66	52.56	**71.42**
	Error Rate	75.34	47.34	**28.38**
	Number of wins	4097	8765	**11839**
n_episodes = 1000	Accuracy	25.34	57.9	**77.19**
	Error Rate	74.71	42.13	**22.64**
	Number of wins	8422	19358	**25569**

The random agent's performance served as a critical control, ensuring that the improvements observed in the WCST NS-DQN agent were due to its architectural enhancements and not random chance.

These results highlight the NS-DQN agent's superior ability to navigate the complex, non-stationary landscape of the WCST environment. Its performance accentuates the importance of architectural considerations in the development of RL agents for dynamic problem spaces.

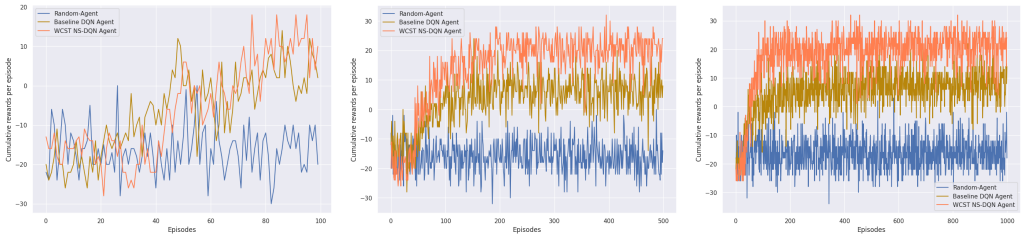

Figure 6: The performance of the three agents over 100, 500 & 1000 episodes in the non-stationary environment

The table 1 provides a comparative analysis of three different agents in terms of accuracy, error rate, and the number of wins over a series of episodes on the Wisconsin Card Sorting Test (WCST). It showcases the performance evolution as the number of episodes reflecting the agent's experience increases. For a baseline comparison, a random agent is used, which presumably selects actions without learning or strategy, resulting in low accuracy and a high error rate across all episodes. This agent's performance is static and does not improve with more episodes, as evidenced by the relatively consistent accuracy and error rates. The Baseline DQN Agent shows a marked improvement over the random agent. As the number of episodes increases, there is a notable improvement in accuracy and a reduction in the error rate. This progression suggests that the DQN agent is learning from its experience, refining its policy to improve its decision-making process. The WCST NS (Non-Stationary) - Agent, which is the focus of the study, shows a superior performance to both the random agent and the Baseline DQN Agent. Starting at 100 episodes, it already surpasses the Baseline DQN Agent with a higher accuracy and lower error rate. This trend continues as the number of episodes grows, indicating that the NS-DQN agent is effectively adapting to the non-stationary environment and learning more efficiently. By 1000 episodes, the NS-DQN agent's accuracy exceeds 77%, with an error rate that has decreased to approximately 22%, substantially outperforming the baseline agents.

The consistent improvement in accuracy and reduction in error rate, coupled with an increasing number of wins, demonstrate that the NS-DQN agent's learning mechanism is more suited to the complexities of the WCST environment. Its capability to adapt to changing rules without explicit notification significantly enhances its performance, highlighting the efficacy of incorporating non-stationarity into the learning algorithm.

6 Conclusion

In conclusion, our study presented a novel approach to the Wisconsin Card Sorting Test (WCST) through the Non-Stationary Deep Q-Network (NS-DQN). This method's ability to integrate the changing rule context directly into the state space represents a significant advancement in handling non-stationary environments. The enhanced DQN model not only demonstrates superior adaptability and learning efficiency but also shows remarkable improvements in decision-making capabilities. The comparative analysis between the NS-DQN agent, a baseline DQN agent, and a random agent across a spectrum of episodes illustrates the effectiveness of our approach. The NS-DQN agent consistently outperformed the baseline DQN agent with higher accuracy, lower error rates, and a greater number of wins. These results underscore the potential of NS-DQN in environments where adaptability to dynamic changes is crucial.

For future work, we aim to explore the integration of other RL strategies and learning models to further enhance the agent's performance. Additionally, the application of NS-DQN to other non-stationary problems and complex environments could be investigated. We also plan to delve into multi-agent scenarios where each agent must adapt not only to the environment but also to the strategies of other agents. Further research may also include exploring meta-learning techniques to enable the agent to learn the process of adapting to changes, rather than adapting to changes themselves. Another interesting avenue is the examination of different ways to encode the rule changes into the state space, perhaps in a more abstract or generalized form that can be applied to a broader range of tasks. In essence, the work presented here opens new pathways for developing more robust and versatile RL agents capable of thriving in the ever-changing landscapes of real-world problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This study was supported by the Ensitech USA LLC - Tecnologico de Monterrey.

References

[1] Pauline Bock and Frédéric Alexandre. [re] the wisconsin card sorting test: Theoretical analysis and modeling in a neuronal network. *The ReScience journal*, 3(1):5, 2019.

[2] Robert K Heaton. Wisconsin card sorting test. *Psychological assessment resources*, 1981.

[3] Bruno Kopp, Bilal Al-Hafez, and Alexander Steinke. Habits, goals, and behavioral signs of cognitive perseveration on wisconsin card-sorting tasks. *Brain Sciences*, 13(6):919, 2023.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[6] Stanislas Dehaene and Jean-Pierre Changeux. The wisconsin card sorting test: Theoretical analysis and modeling in a neuronal network. *Cerebral cortex*, 1(1):62–79, 1991.

[7] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 17(73):1–29, 2016.

[8] Leslie Pack Kaelbling. *Learning in embedded systems*. MIT press, 1993.

[9] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.

[10] Sebastian Tschiatschek, Kai Arulkumaran, Jan Stühmer, and Katja Hofmann. Variational inference for data-efficient model learning in pomdps. *arXiv preprint arXiv:1805.09281*, 2018.

[11] Alexander Steinke, Florian Lange, and Bruno Kopp. A multi-level reinforcement-learning model of wisconsin card sorting test performance. In *2019 Conference on Cognitive Computational Neuroscience*, volume 10, pages 2019–1030, 2019.

[12] Yuzhen Qin, Tommaso Menara, Samet Oymak, ShiNung Ching, and Fabio Pasqualetti. Non-stationary representation learning in sequential linear bandits. *IEEE Open Journal of Control Systems*, 1:41–56, 2022.

[13] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. *Robotica*, 17(2):229–235, 1999.

[14] Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, 2020.

[15] Xiangtian Deng, Yi Zhang, and He Qi. Towards optimal hvac control in non-stationary building environments combining active change detection and deep reinforcement learning. *Building and environment*, 211:108680, 2022.

[16] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

[17] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in neural information processing systems*, 19, 2006.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[19] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[20] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.